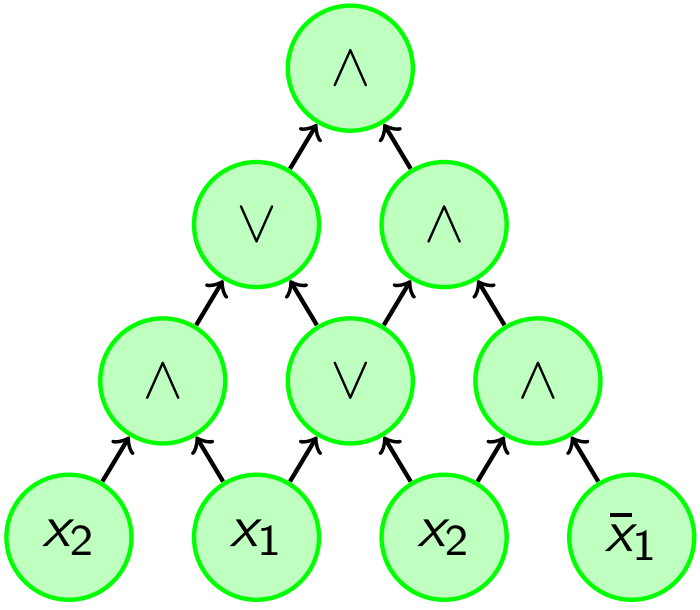
Pebble Games and Complexity

Siu Man Chan

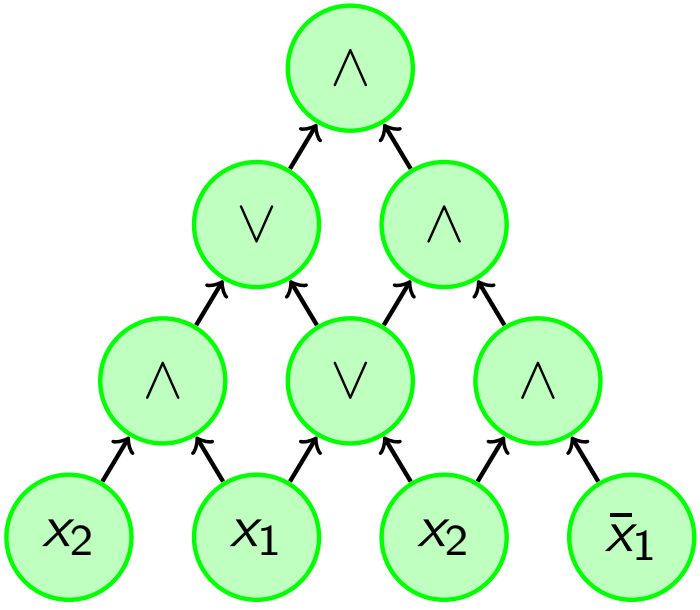
Princeton CCI

21 Jan, 2014 @ IAS

Circuit Evaluation Problem

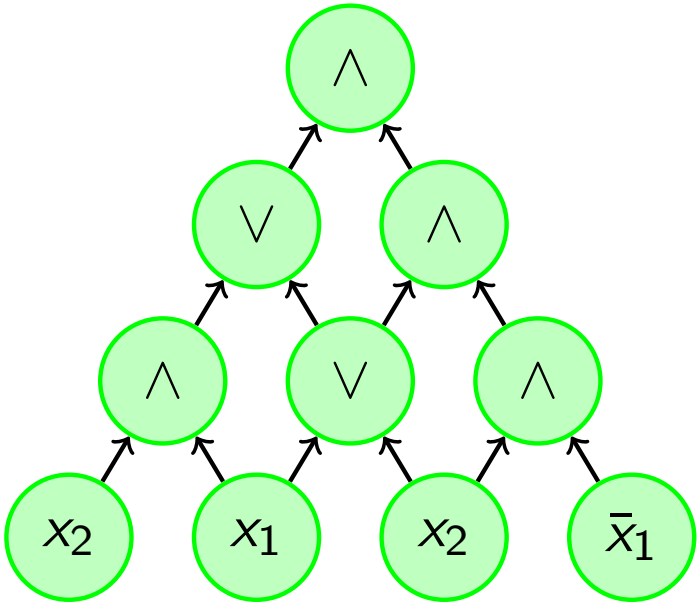


Circuit Evaluation Problem



- Input:
  - circuit C
  - instance x

Circuit Evaluation Problem



- Input:
  circuit C
  instance x
- Output:
  Result of evaluating C on x

Circuit Evaluation Problem

- Complete for P

Circuit Evaluation Problem

- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

Circuit Evaluation Problem

- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

$$\mathsf{NC}^i = \qquad \text{size } n^{O(1)}, \qquad \text{depth } O(\log^i n)$$

Circuit Evaluation Problem

- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

$$\begin{aligned} \mathsf{NC}^i = & \quad \text{size } n^{O(1)}, & \text{depth } O(\log^i n) \\ \approx & \quad \text{processors } n^{O(1)}, & \text{parallel time } O(\log^i n) \end{aligned}$$

Circuit Evaluation Problem

- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

$$\begin{aligned} \mathsf{NC}^i = & \quad \text{size } n^{O(1)}, & \text{depth } O(\log^i n) \\ \approx & \quad \text{processors } n^{O(1)}, & \text{parallel time } O(\log^i n) \end{aligned}$$

$$\mathsf{NC}^1 \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

Circuit Evaluation Problem

- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

$$\begin{aligned} \mathsf{NC}^i = & \quad \text{size } n^{O(1)}, & \text{depth } O(\log^i n) \\ \approx & \quad \text{processors } n^{O(1)}, & \text{parallel time } O(\log^i n) \end{aligned}$$

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

Circuit Evaluation Problem

- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

$$\begin{aligned} \mathsf{NC}^i = & \quad \text{size } n^{O(1)}, & \text{depth } O(\log^i n) \\ \approx & \quad \text{processors } n^{O(1)}, & \text{parallel time } O(\log^i n) \end{aligned}$$

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

- Concerns space and parallel complexity

Circuit Evaluation Problem

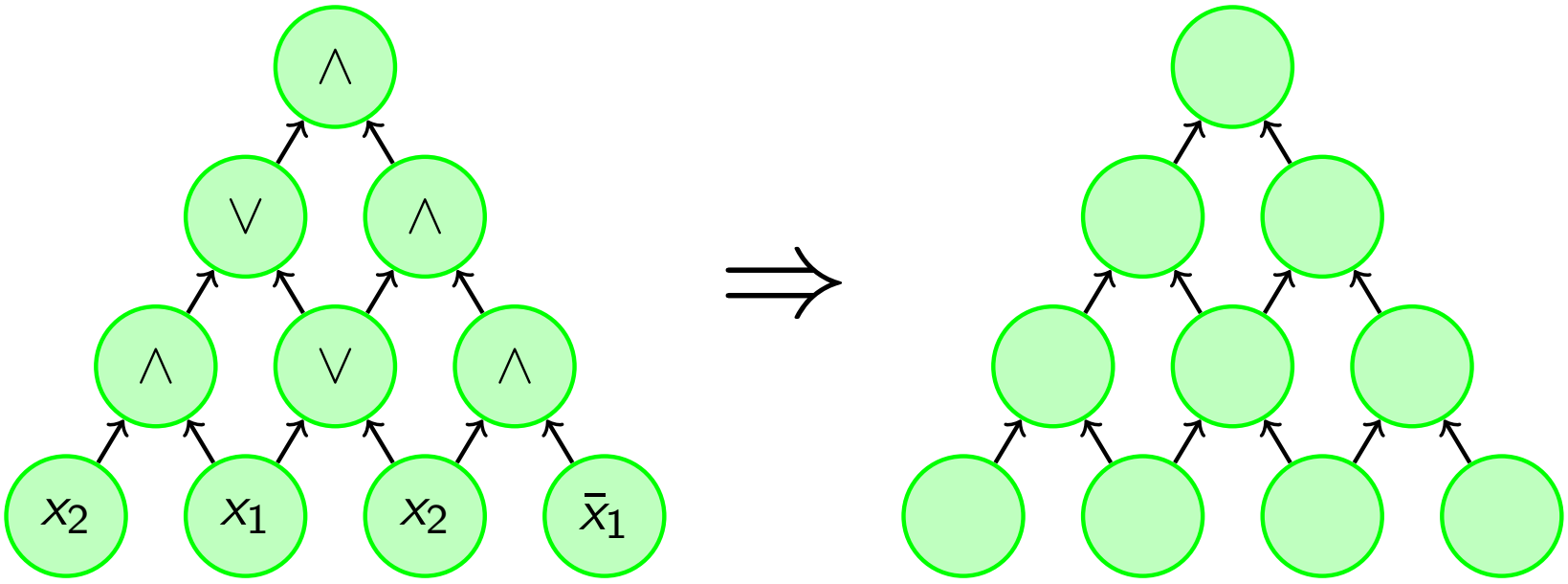
- Complete for P
- "Complete" for NC^i for restricted C of depth $\leq O(\log^i n)$

$$\begin{aligned} \mathsf{NC}^i = & \quad \text{size } n^{O(1)}, & \text{depth } O(\log^i n) \\ \approx & \quad \text{processors } n^{O(1)}, & \text{parallel time } O(\log^i n) \end{aligned}$$

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

- Concerns space and parallel complexity
- Applications: Database query algorithms, Data flow models, Big Data computation, etc.

Black Pebble Game



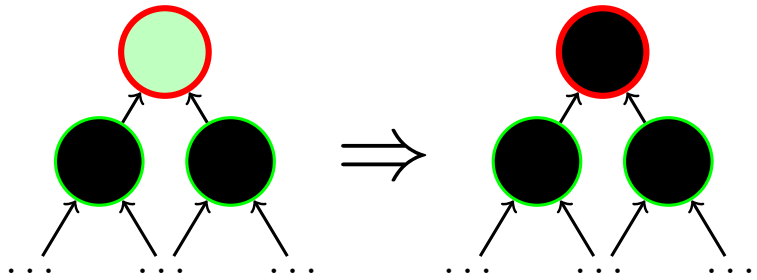
Black Pebble Game

Algorithms for Circuit Evaluation

Black Pebble Game

Algorithms for Circuit Evaluation

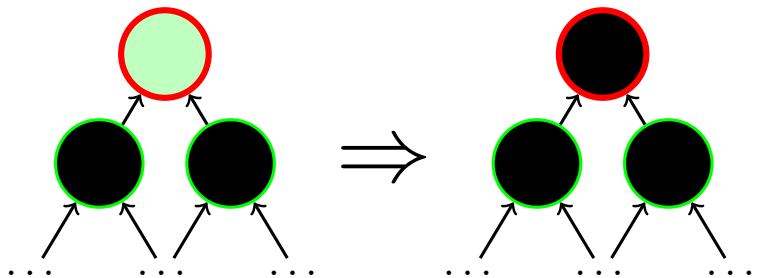
- Rule 1: add pebble to v if all immediate predecessors of v are pebbled

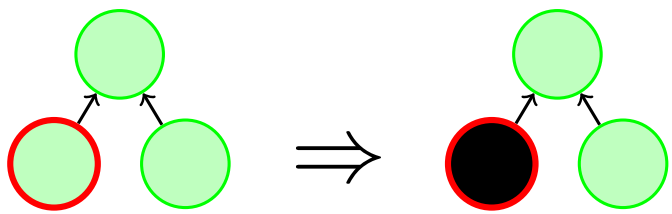


Black Pebble Game

Algorithms for Circuit Evaluation

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled

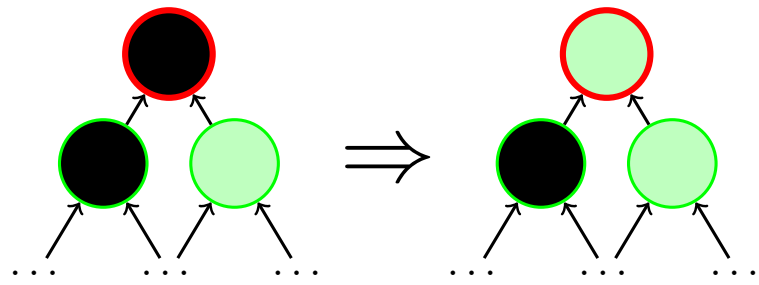




Black Pebble Game

Algorithms for Circuit Evaluation

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time



Black Pebble Game

Algorithms for Circuit Evaluation

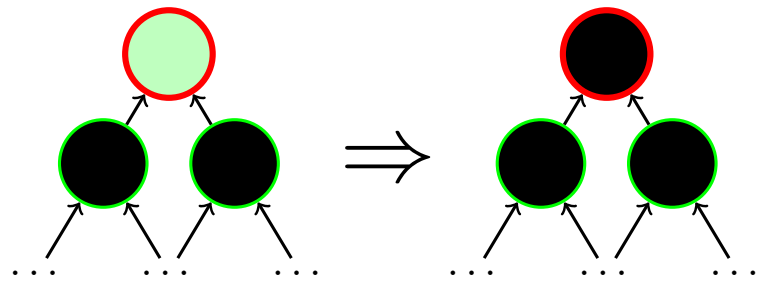
- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time

- Pebbled means value stored in memory

Black Pebble Game

Algorithms for Circuit Evaluation

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time

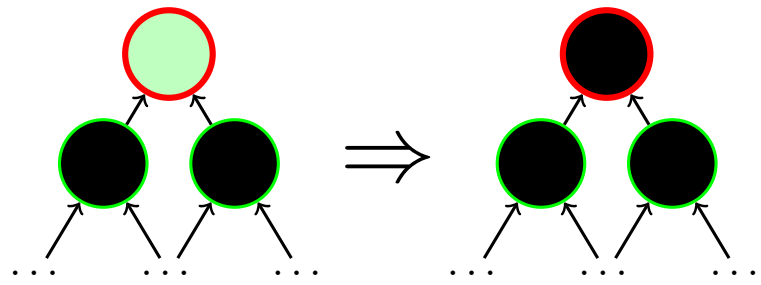


- Pebbled means value stored in memory

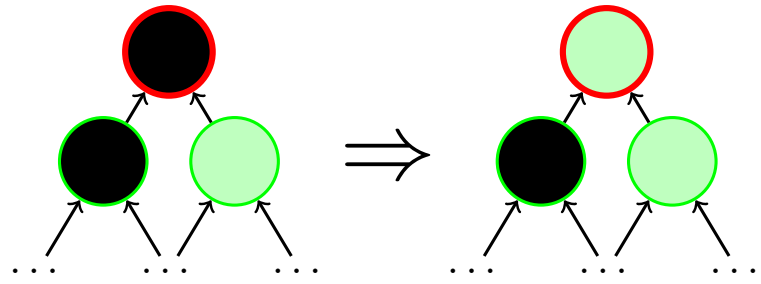
Black Pebble Game

Algorithms for Circuit Evaluation

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time



- Pebbled means value stored in memory



Black Pebble Game

Algorithms for Circuit Evaluation

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time

- Pebbled means value stored in memory

Space needed $\leq O(\#\text{Pebbles})$

Black Pebble Game

Algorithms for Circuit Evaluation

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time

$$\forall G$$
$$\#\text{Pebbles(G)} \leq O\big(|G|/\log|G|\big)$$

[Hopcroft–Paul–Valiant '77]

- Pebbled means value stored in memory

$$\text{Space needed} \leq O(\#\text{Pebbles})$$

Black Pebble Game

Algorithms for Circuit Evaluation

$$\mathrm{Time}[t] \subseteq \mathrm{Space}[t/\log t]$$

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble at any time

$$\forall G$$
$$\#\mathrm{Pebbles(G)} \leq O\big(|G|/\log|G|\big)$$

[Hopcroft–Paul–Valiant '77]

- Pebbled means value stored in memory

$$\text{Space needed} \leq O(\#\mathrm{Pebbles})$$

Lower Bounds by Pebble Games

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

m-$\mathsf{NC}^1 \subsetneq$ m-NL m-circuits [Karchmer–Wigderson '90]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]
$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Raz–McKenzie '99]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[C.–Potechin '12]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]
$\text{m-L} \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-switching-networks	[C.–Potechin '12]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]
$\text{m-L} \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-switching-networks	[C.–Potechin '12]
$\text{sem-NC}^i \subsetneq \text{sem-NC}^{i+1}$ $\text{sem-NC} \subsetneq \text{sem-P}$	sem-circuits	[C. '13]

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]	
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]	Raz–McKenzie pebble game
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]	Reversible pebble game
$\text{m-L} \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-switching-networks	[C.–Potechin '12]	Reversible pebble game
$\text{sem-NC}^i \subsetneq \text{sem-NC}^{i+1}$ $\text{sem-NC} \subsetneq \text{sem-P}$	sem-circuits	[C. '13]	Dymond–Tompa pebble game Raz–McKenzie pebble game

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]	
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]	Raz–McKenzie pebble game
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]	Reversible pebble game
$\text{m-L} \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-switching-networks	[C.–Potechin '12]	Reversible pebble game
$\text{sem-NC}^i \subsetneq \text{sem-NC}^{i+1}$ $\text{sem-NC} \subsetneq \text{sem-P}$	sem-circuits	[C. '13]	Dymond–Tompa pebble game Raz–McKenzie pebble game

Theorem ([C. '13])

$\forall G$ *Simulation of strategies among*
Raz–McKenzie game, reversible game, and Dymond–Tompa game.

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-NL}$	m-circuits	[Karchmer–Wigderson '90]	
$\text{m-NC}^1 \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]	Raz–McKenzie pebble game
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]	Reversible pebble game
$\text{m-L} \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-switching-networks	[C.–Potechin '12] [Filmus–Pitassi–Robere–Cook '13]	Reversible pebble game
$\text{sem-NC}^i \subsetneq \text{sem-NC}^{i+1}$ $\text{sem-NC} \subsetneq \text{sem-P}$	sem-circuits	[C. '13]	Dymond–Tompa pebble game Raz–McKenzie pebble game

Theorem ([C. '13])

$\forall G$ *Simulation of strategies among*
Raz–McKenzie game, reversible game, and Dymond–Tompa game.

Lower Bounds by Pebble Games

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC}^3 \subseteq \cdots \subseteq \mathsf{NC} \subseteq \mathsf{P}$$

$\text{m-NC}^1 \subsetneq \text{m-L}$	m-circuits	[Karchmer–Wigderson '90]	
$\text{m-NC}^1 \subsetneq \text{m-L}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-circuits	[Raz–McKenzie '99]	Raz–McKenzie pebble game
$\text{m-L} \subsetneq \text{m-NL}$	m-switching-networks	[Potechin '10]	Reversible pebble game
$\text{m-L} \subsetneq \text{m-NL}$ $\text{m-NC}^i \subsetneq \text{m-NC}^{i+1}$ $\text{m-NC} \subsetneq \text{m-P}$	m-switching-networks	[C.–Potechin '12] [Filmus–Pitassi–Robere–Cook '13]	Reversible pebble game
$\text{sem-NC}^i \subsetneq \text{sem-NC}^{i+1}$ $\text{sem-NC} \subsetneq \text{sem-P}$	sem-circuits	[C. '13]	Dymond–Tompa pebble game Raz–McKenzie pebble game

Theorem ([C. '13])

$\forall G$ *Simulation of strategies among*
Raz–McKenzie game, reversible game, and Dymond–Tompa game.

Semantic Circuit Bounds

Theorem (Karchmer–Wigderson)

Circuit Depth = Communication Complexity

Semantic Circuit Bounds

Theorem (Karchmer–Wigderson)

Circuit Depth = Communication Complexity

NC^1 vs NC^2 Universal composition relation [Edmonds–Impagliazzo–Rudich–Sgall '01]

Semantic Circuit Bounds

Theorem (Karchmer–Wigderson)

Circuit Depth = Communication Complexity

NC^1 vs NC^2	Universal composition relation	[Edmonds–Impagliazzo–Rudich–Sgall '01]
NC^1 vs NC^2	Universal composition relation	[Håstad–Wigderson '97]

Semantic Circuit Bounds

Theorem (Karchmer–Wigderson)

Circuit Depth = Communication Complexity

NC^1 vs NC^2	Universal composition relation	[Edmonds–Impagliazzo–Rudich–Sgall '01]
NC^1 vs NC^2	Universal composition relation	[Håstad–Wigderson '97]
NC^i vs NC^{i+1} NC vs P	Iterated indexing	[C. '13]

Semantic Circuit Bounds

Theorem (Karchmer–Wigderson)

Circuit Depth = Communication Complexity

NC^1 vs NC^2	Universal composition relation	[Edmonds–Impagliazzo–Rudich–Sgall '01]
NC^1 vs NC^2	Universal composition relation	[Håstad–Wigderson '97]
NC^i vs NC^{i+1} NC vs P	Iterated indexing	[C. '13]

In the bounds for Iterated indexing:

- Upper bound by Dymond–Tompa game
- Lower bound by Raz–McKenzie game

Semantic Circuit Bounds

Theorem (Karchmer–Wigderson)

Circuit Depth = Communication Complexity

NC^1 vs NC^2	Universal composition relation	[Edmonds–Impagliazzo–Rudich–Sgall '01] [Gavinsky–Meir–Weinstein–Wigderson '13]
NC^1 vs NC^2	Universal composition relation	[Håstad–Wigderson '97]
NC^i vs NC^{i+1} NC vs P	Iterated indexing	[C. '13]

In the bounds for Iterated indexing:

- Upper bound by Dymond–Tompa game
- Lower bound by Raz–McKenzie game

My audience

- Parallel Complexity
- Space Complexity
- Randomised Complexity
- Communication Complexity
- Decision Tree Complexity (Certificate Complexity)
- Proof Complexity
- Algebraic Complexity

Reversible game

$\|$

Dymond–Tompa game

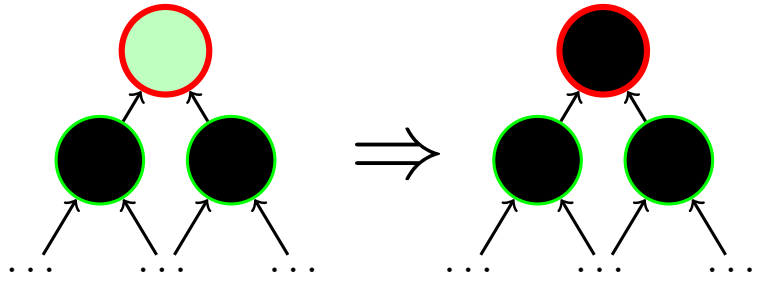
$\|$

Raz–McKenzie game

Reversible Pebble Game

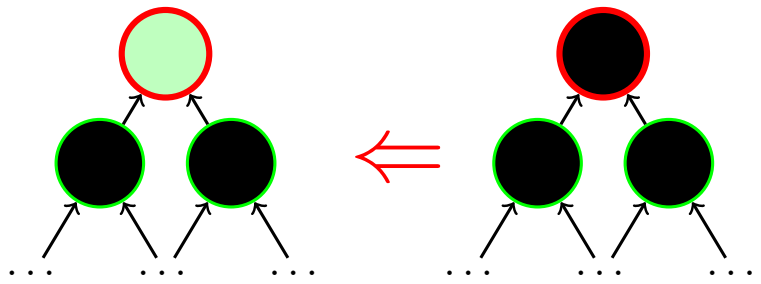
Reversible Pebble Game

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled



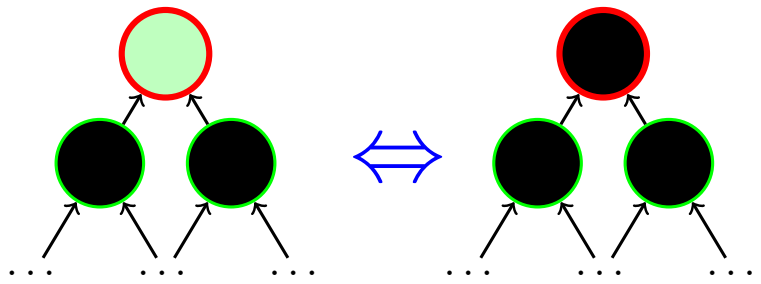
Reversible Pebble Game

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble if all immediate predecessors of v are pebbled



Reversible Pebble Game

- Rule 1: add pebble to v if all immediate predecessors of v are pebbled
- Rule 2: remove pebble if all immediate predecessors of v are pebbled



Reversible Pebble Game

- Reversible Computation [Bennett '73]:

Reversible Pebble Game

- Reversible Computation [Bennett '73]:
  - May reduce energy dissipation

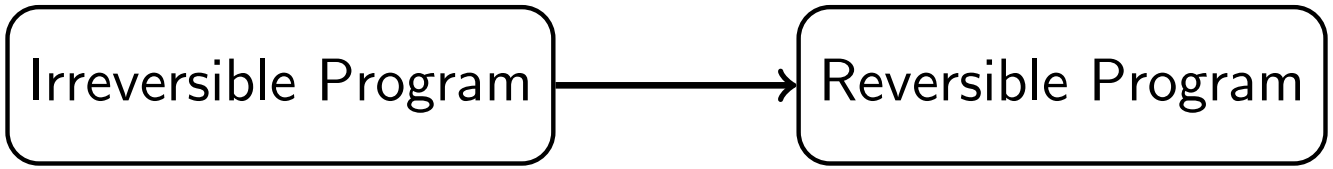
Reversible Pebble Game

- Reversible Computation [Bennett '73]:
  - May reduce energy dissipation
  - Observation-free quantum computation is reversible

Reversible Pebble Game

- Reversible Computation [Bennett '73]:
  - May reduce energy dissipation
  - Observation-free quantum computation is reversible
  - Reversible simulation of irreversible computation [Bennett '89]

    [Li–Vitanyi '96, '97] [Král'ovič '01]



Reversible Pebble Game

- Reversible Computation [Bennett '73]:
  - May reduce energy dissipation
  - Observation-free quantum computation is reversible
  - Reversible simulation of irreversible computation [Bennett '89]

    [Li–Vitanyi '96, '97] [Král'ovič '01]
- Monotone space lower bounds [Potechin '10] [C.–Potechin '12]:

Reversible Pebble Game

- Reversible Computation [Bennett '73]:
  - May reduce energy dissipation
  - Observation-free quantum computation is reversible
  - Reversible simulation of irreversible computation [Bennett '89]

    [Li–Vitanyi '96, '97] [Král'ovič '01]
- Monotone space lower bounds [Potechin '10] [C.–Potechin '12]:
  - Determinism equals reversibility/symmetry [Lange–McKenzie–Tapp '00]

    [Reingold '08]

Dymond–Tompa Pebble Game

Dymond–Tompa Pebble Game

- **To design parallel algorithms** [Dymond–Tompa '85, Gál–Jang '11]

Dymond–Tompa Pebble Game

- **To design parallel algorithms** [Dymond–Tompa '85, Gál–Jang '11]
  Give parallel speed-ups (when #processors is unbounded).

Dymond–Tompa Pebble Game

- **To design parallel algorithms** [Dymond–Tompa '85, Gál–Jang '11]
  Give parallel speed-ups (when #processors is unbounded).
- **Capture complexity classes and inclusions** [Venkateswaran–Tompa '89]

Dymond–Tompa Pebble Game

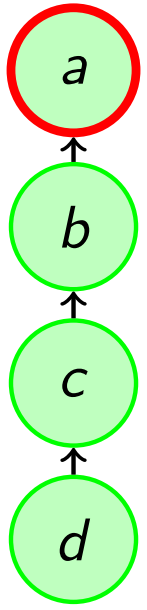
- **To design parallel algorithms** [Dymond–Tompa '85, Gál–Jang '11]
  Give parallel speed-ups (when #processors is unbounded).
- **Capture complexity classes and inclusions** [Venkateswaran–Tompa '89]
  - (#Pebble used) characterizes parallelism in $\mathsf{NC}^i, \mathsf{NC}, \mathsf{P}$, etc;

Dymond–Tompa Pebble Game

- **To design parallel algorithms** [Dymond–Tompa '85, Gál–Jang '11]
  Give parallel speed-ups (when #processors is unbounded).
- **Capture complexity classes and inclusions** [Venkateswaran–Tompa '89]
  - (#Pebble used) characterizes parallelism in $\mathsf{NC}^i, \mathsf{NC}, \mathsf{P}$, etc;
  - Simulates the inclusion of $\mathsf{NL} \subseteq \mathsf{NC}^2$.

Parallel Evaluation, Recursively

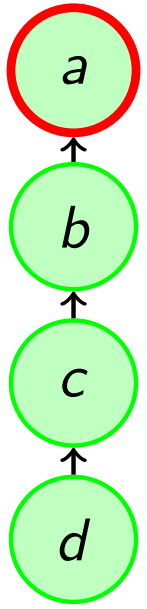
To compute the value at a



Parallel Evaluation, Recursively

To compute the value at a

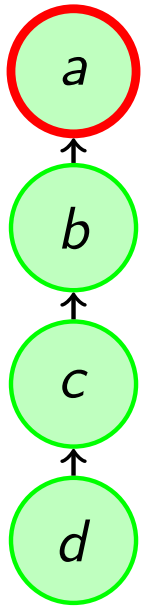
1. Pick a node, say c



Parallel Evaluation, Recursively

To compute the value at a

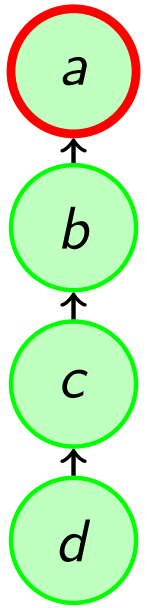
1. Pick a node, say c
2. In parallel, do



Parallel Evaluation, Recursively

To compute the value at a

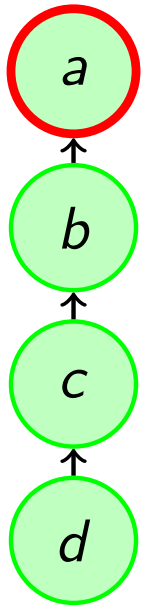
1. Pick a node, say c
2. In parallel, do
   - Compute the value at c



Parallel Evaluation, Recursively

To compute the value at a

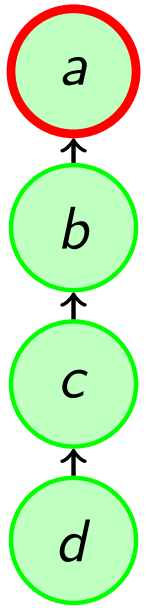
1. Pick a node, say c
2. In parallel, do
   - Compute the value at c
   - For each possible value v_c of c, assume v_c is correct and compute the value at a



Parallel Evaluation, Recursively

To compute the value at a

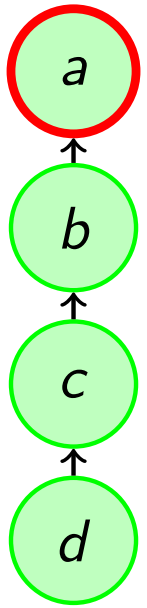
1. Pick a node, say c
2. In parallel, do
   - Compute the value at c
   - For each possible value v_c of c, assume v_c is correct and compute the value at a
3. Recurse!



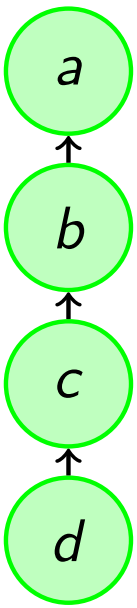
Parallel Evaluation, Recursively

To compute the value at a

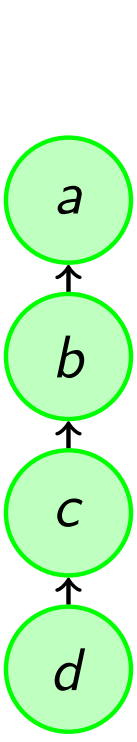
1. Pick a node, say c
2. In parallel, do
   - Compute the value at c
   - For each possible value v_c of c, assume v_c is correct and compute the value at a
3. Recurse!
4. Combine the results in Step 2 in constant time

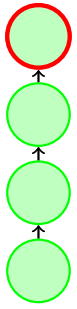


Parallel Evaluation, Recursively

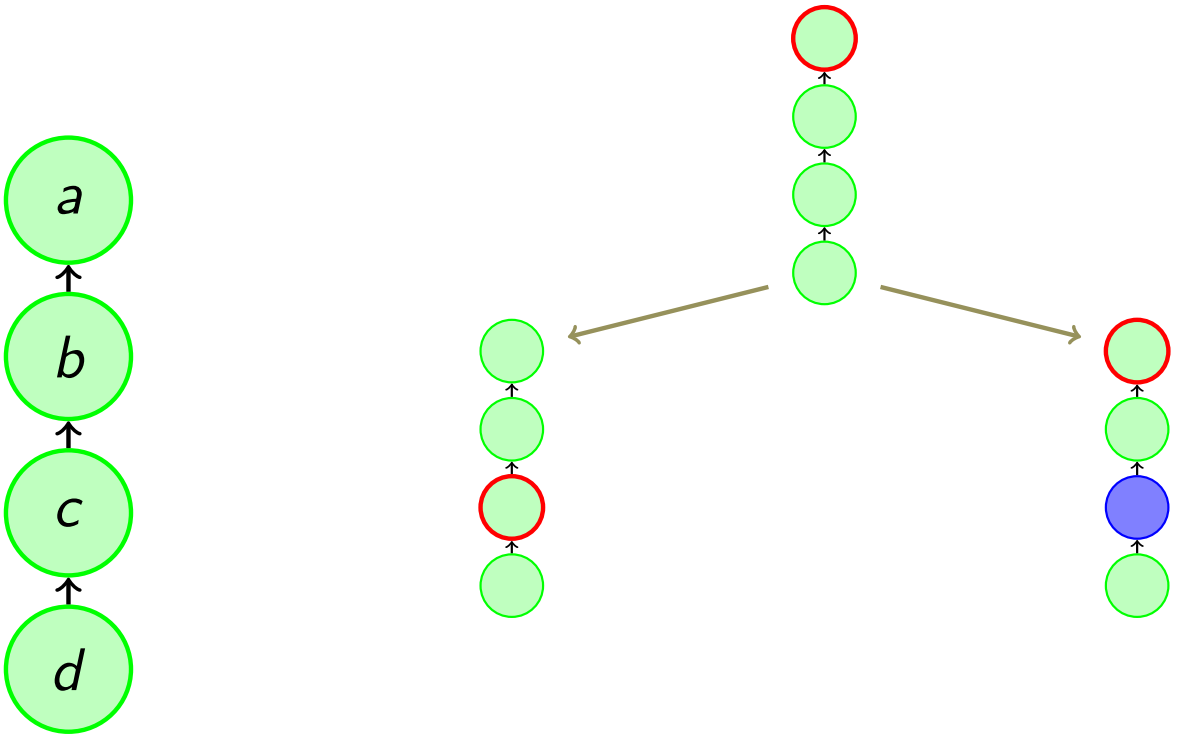


Parallel Evaluation, Recursively

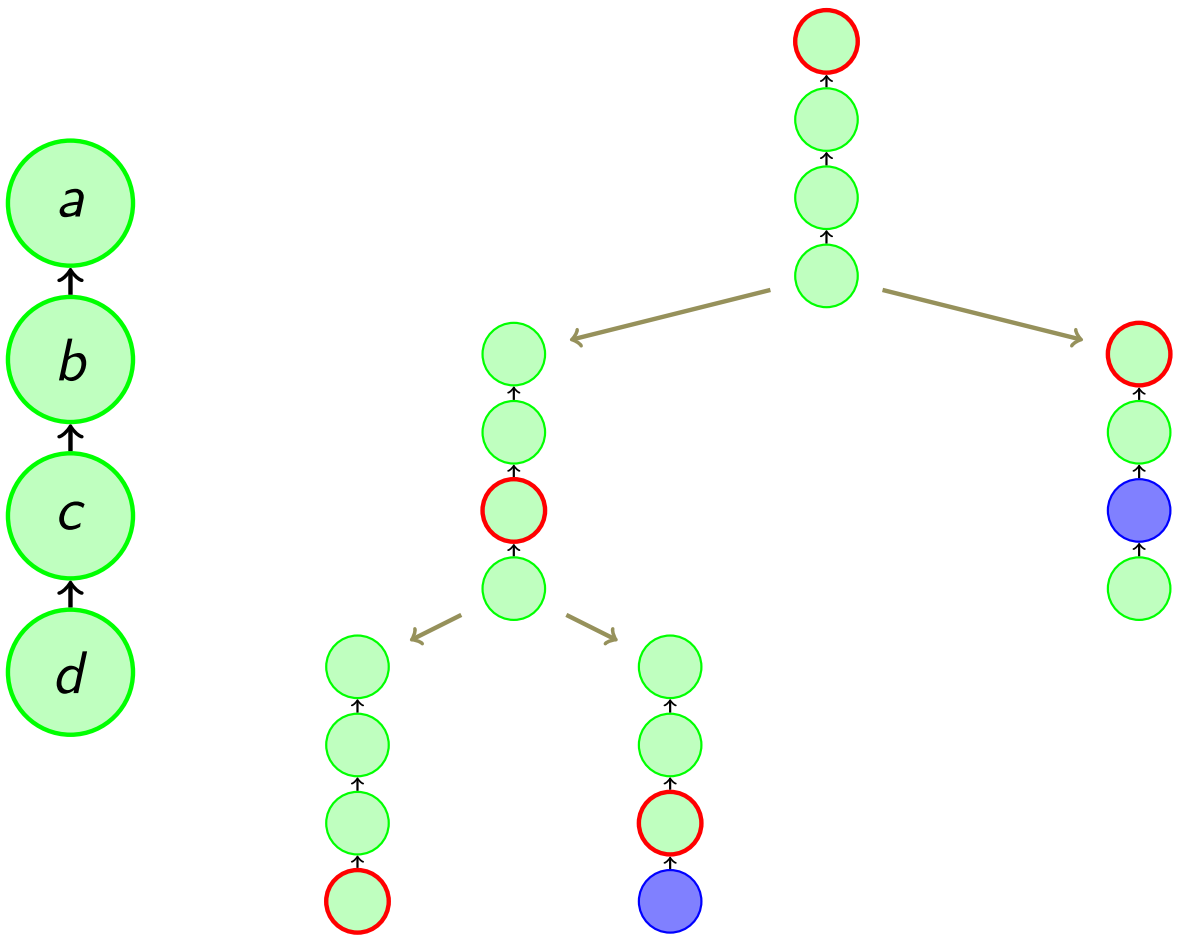




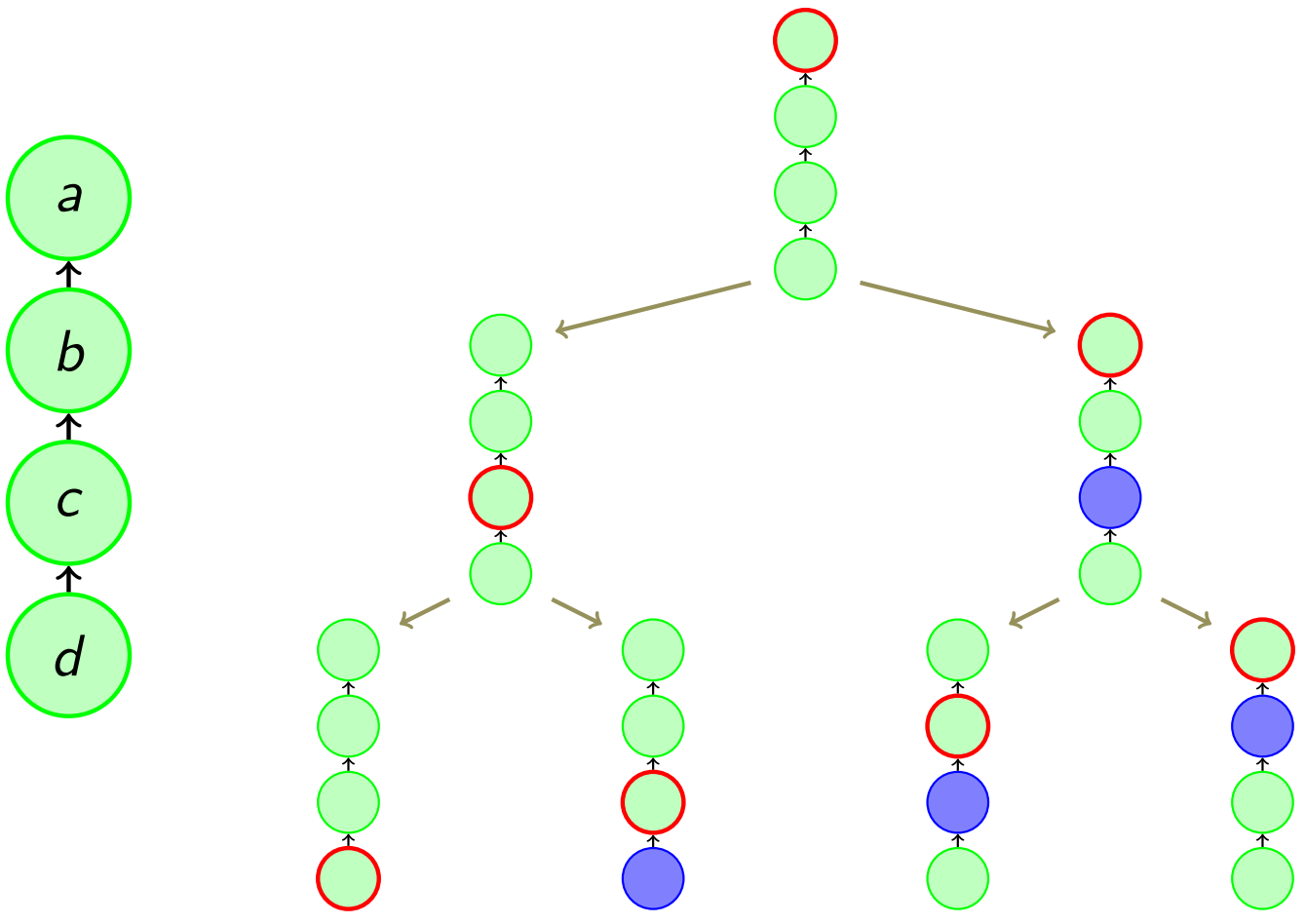
Parallel Evaluation, Recursively



Parallel Evaluation, Recursively



Parallel Evaluation, Recursively



Dymond–Tompa Game

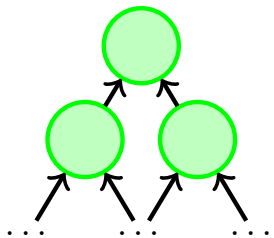
- Two players: **Pebbler** and **Challenger**, competitive

Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first

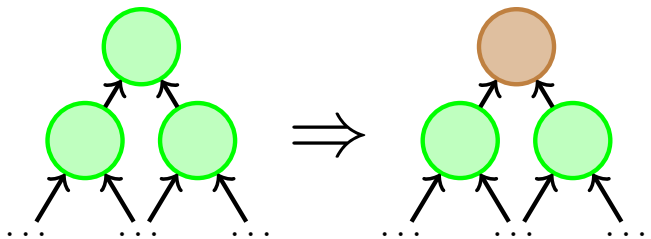
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:



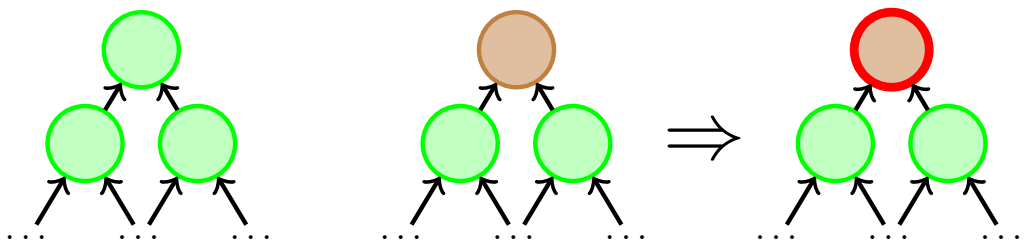
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink



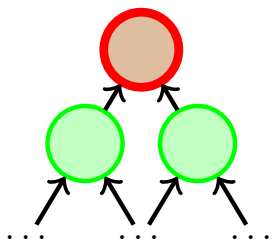
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink (exactly one pebbled node is challenged any time)



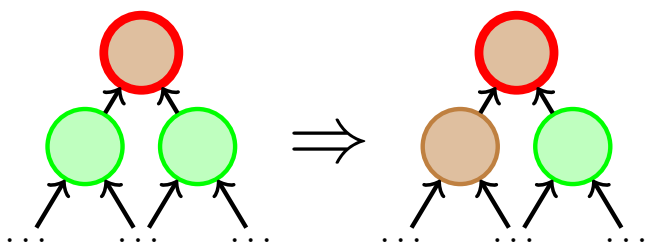
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink (exactly one pebbled node is challenged any time)
- Each round:



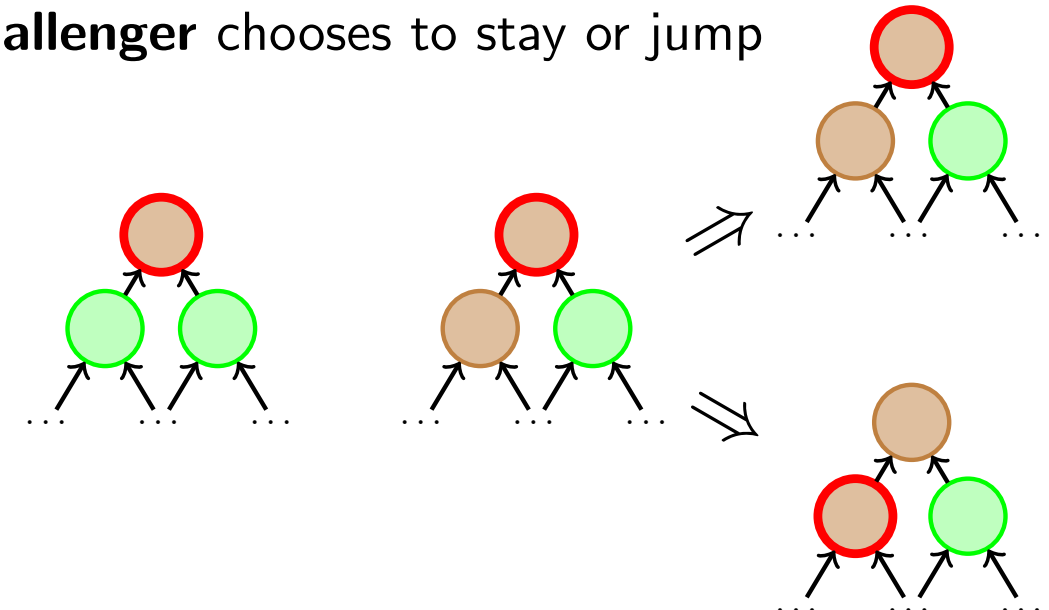
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink (exactly one pebbled node is challenged any time)
- Each round:
  - **Pebbler** chooses a node to pebble



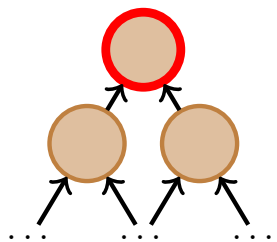
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink (exactly one pebbled node is challenged any time)
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Challenger** chooses to stay or jump



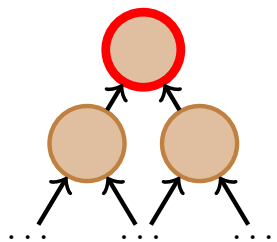
Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink (exactly one pebbled node is challenged any time)
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Challenger** chooses to stay or jump
- **Pebbler** wins if, before she moves, the challenged node has all immediate predecessors pebbled



Dymond–Tompa Game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink (exactly one pebbled node is challenged any time)
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Challenger** chooses to stay or jump
- **Pebbler** wins if, before she moves, the challenged node has all immediate predecessors pebbled
- **Challenger** aims to delay the inevitable



Raz–McKenzie Pebble Game

Raz–McKenzie Pebble Game

- **Give depth lower bounds to monotone circuits** [Raz–McKenzie '99]

Raz–McKenzie Pebble Game

- **Give depth lower bounds to monotone circuits** [Raz–McKenzie '99]
  Motivated by decision tree complexity of search problems

  [Lovász–Naor–Newman–Wigderson '95]

Raz–McKenzie Pebble Game

- **Give depth lower bounds to monotone circuits** [Raz–McKenzie '99]
  Motivated by decision tree complexity of search problems
  [Lovász–Naor–Newman–Wigderson '95]
- **Applications to Proof Complexity**:
  - Inspired pebbling contradictions *(next slide)*

Raz–McKenzie Pebble Game

- **Give depth lower bounds to monotone circuits** [Raz–McKenzie '99]
  Motivated by decision tree complexity of search problems
  [Lovász–Naor–Newman–Wigderson '95]
- **Applications to Proof Complexity**:
  - Inspired pebbling contradictions *(next slide)*
  - Separation and Trade-off Results:
    - Cutting plane refutations [Bonet–Esteban–Galesi–Galesi '98]
    - Treelike resolution refutations [Ben-Sasson–Impagliazzo–Wigderson '04]
      [Urquhart '11]
    - Regular resolution refutations [Alekhnovich–Johannsen–Pitassi–Urquhart '07]
    - Clause learning algorithms [Beame–Impagliazzo–Pitassi–Segerlind '10]
    - Nullstellenzatz and Polynomial Calculus
      [Buresh-Oppenheim–Clegg–Impagliazzo–Pitassi '02]
    - k-DNF resolution refutation [Esteban–Galesi–Messner '04]

Raz–McKenzie Pebble Game

- **Give depth lower bounds to monotone circuits** [Raz–McKenzie '99]
  Motivated by decision tree complexity of search problems
  [Lovász–Naor–Newman–Wigderson '95]
- **Applications to Proof Complexity**:
  - Inspired pebbling contradictions *(next slide)*
  - Separation and Trade-off Results:
    - Cutting plane refutations [Bonet–Esteban–Galesi–Galesi '98]
    - Treelike resolution refutations [Ben-Sasson–Impagliazzo–Wigderson '04]
      [Urquhart '11]
    - Regular resolution refutations [Alekhnovich–Johannsen–Pitassi–Urquhart '07]
    - Clause learning algorithms [Beame–Impagliazzo–Pitassi–Segerlind '10]
    - Nullstellenzatz and Polynomial Calculus
      [Buresh-Oppenheim–Clegg–Impagliazzo–Pitassi '02]
    - k-DNF resolution refutation [Esteban–Galesi–Messner '04]
  - Depth of resolution refutation [C. '13]

Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:

Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node

Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are $\textsc{True}$,

Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are $\textsc{True}$,

Implication Truth propagates through the graph,

Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

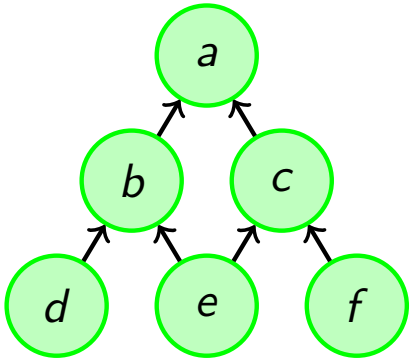
Source	All source variables are $\textsc{True}$,
Implication	Truth propagates through the graph,
Sink	The sink variable is false.

Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are $\textsc{True}$,
Implication Truth propagates through the graph,
Sink The sink variable is false.

Example:



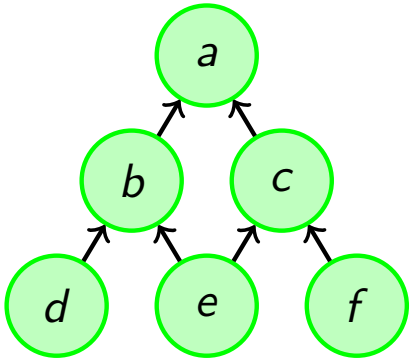
Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are TRUE,
Implication Truth propagates through the graph,
Sink The sink variable is false.

d

Example:



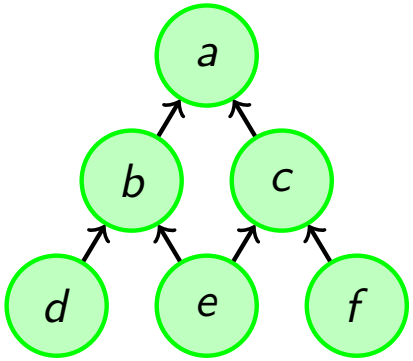
Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are TRUE,
Implication Truth propagates through the graph,
Sink The sink variable is false.

d
e

Example:

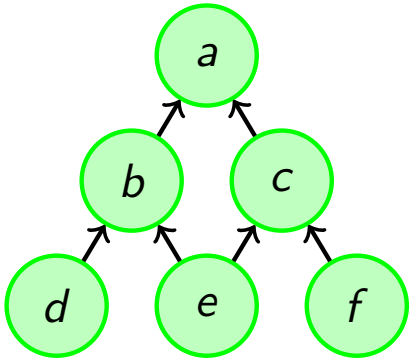


Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are TRUE,
Implication Truth propagates through the graph,
Sink The sink variable is false.

Example: d e f



Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are $\textsc{True}$,
Implication Truth propagates through the graph,
Sink The sink variable is false.

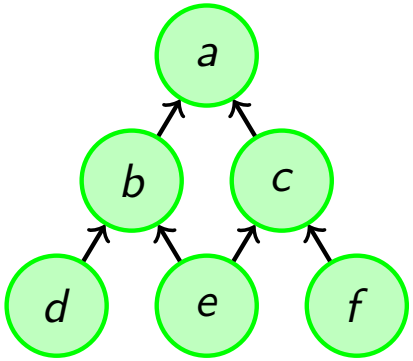
Example:

$d \wedge e \Rightarrow b$ $\quad \bar{d} \vee \bar{e} \vee b$

d

e

f



Pebbling Contradictions

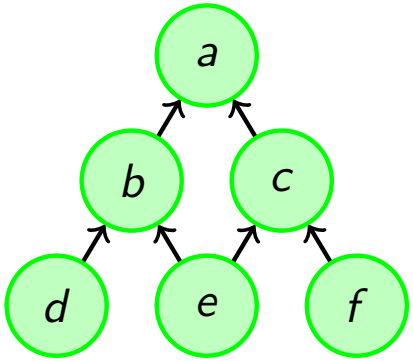
- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are TRUE,
Implication Truth propagates through the graph,
Sink The sink variable is false.

Example:

d
e
f

$d \wedge e \Rightarrow b$ $\quad \bar{d} \vee \bar{e} \vee b$
$e \wedge f \Rightarrow c$ $\quad \bar{e} \vee \bar{f} \vee c$



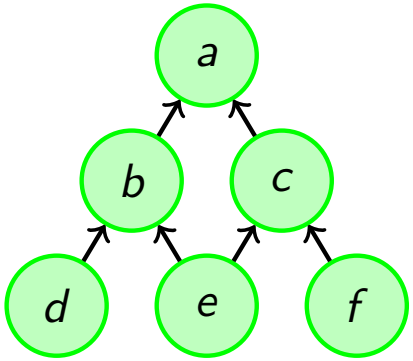
Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are $\textsc{True}$,
Implication Truth propagates through the graph,
Sink The sink variable is false.

Example:

	d
	e
	f
$d \wedge e \Rightarrow b$	$\bar{d} \vee \bar{e} \vee b$
$e \wedge f \Rightarrow c$	$\bar{e} \vee \bar{f} \vee c$
$b \wedge c \Rightarrow a$	$\bar{b} \vee \bar{c} \vee a$



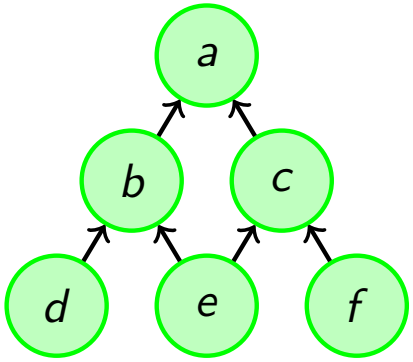
Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source All source variables are $\textsc{True}$,
Implication Truth propagates through the graph,
Sink The sink variable is false.

Example:

	d
	e
	f
$d \wedge e \Rightarrow b$	$\bar{d} \vee \bar{e} \vee b$
$e \wedge f \Rightarrow c$	$\bar{e} \vee \bar{f} \vee c$
$b \wedge c \Rightarrow a$	$\bar{b} \vee \bar{c} \vee a$
	$\bar{a}$



Pebbling Contradictions

- Given G, construct an unsatisfiable CNF Σ_G:
- One variable per node
- Add the following clauses:

Source	All source variables are $\textsc{True}$,
Implication	Truth propagates through the graph,
Sink	The sink variable is false.

Resolution refutation of minimum depth for Σ_G.

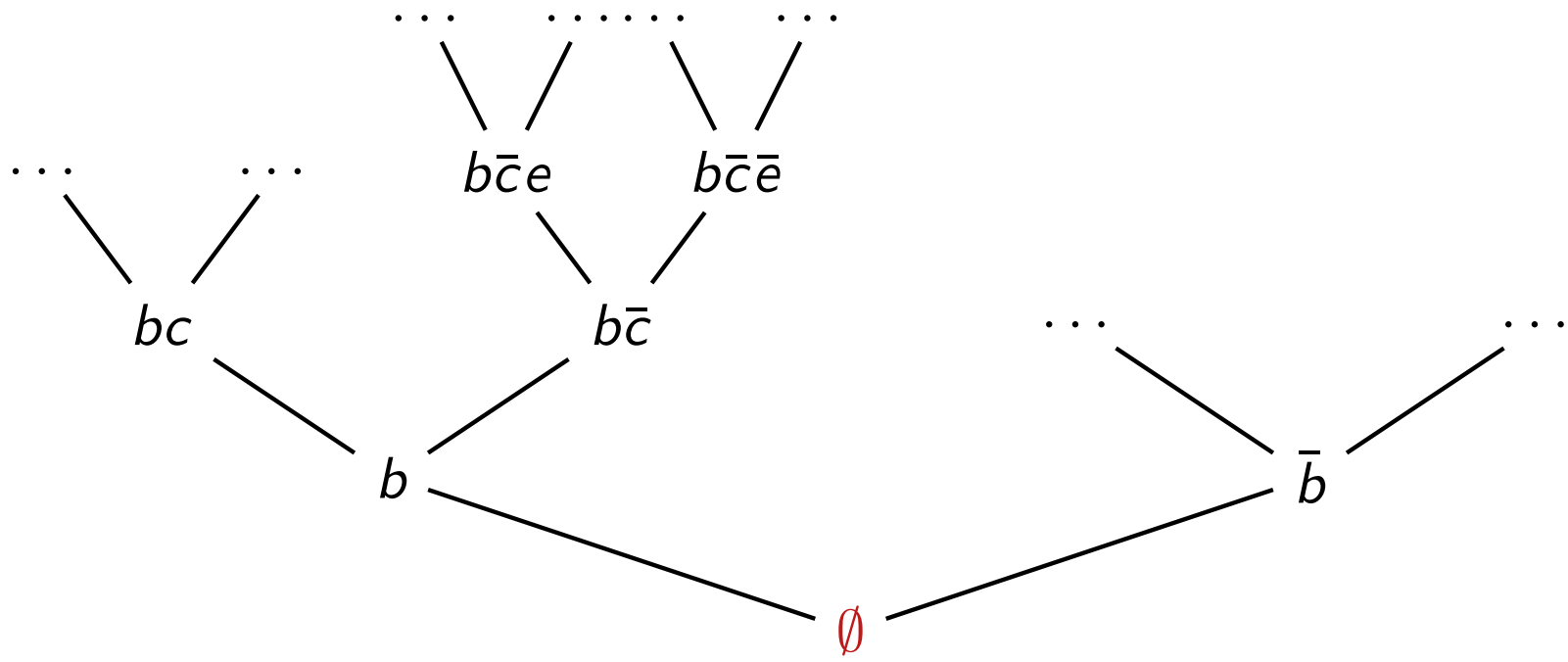
Resolution Refutation

Resolution Refutation

Resolution Step: $\dfrac{A \vee x \quad B \vee \bar{x}}{A \vee B}$

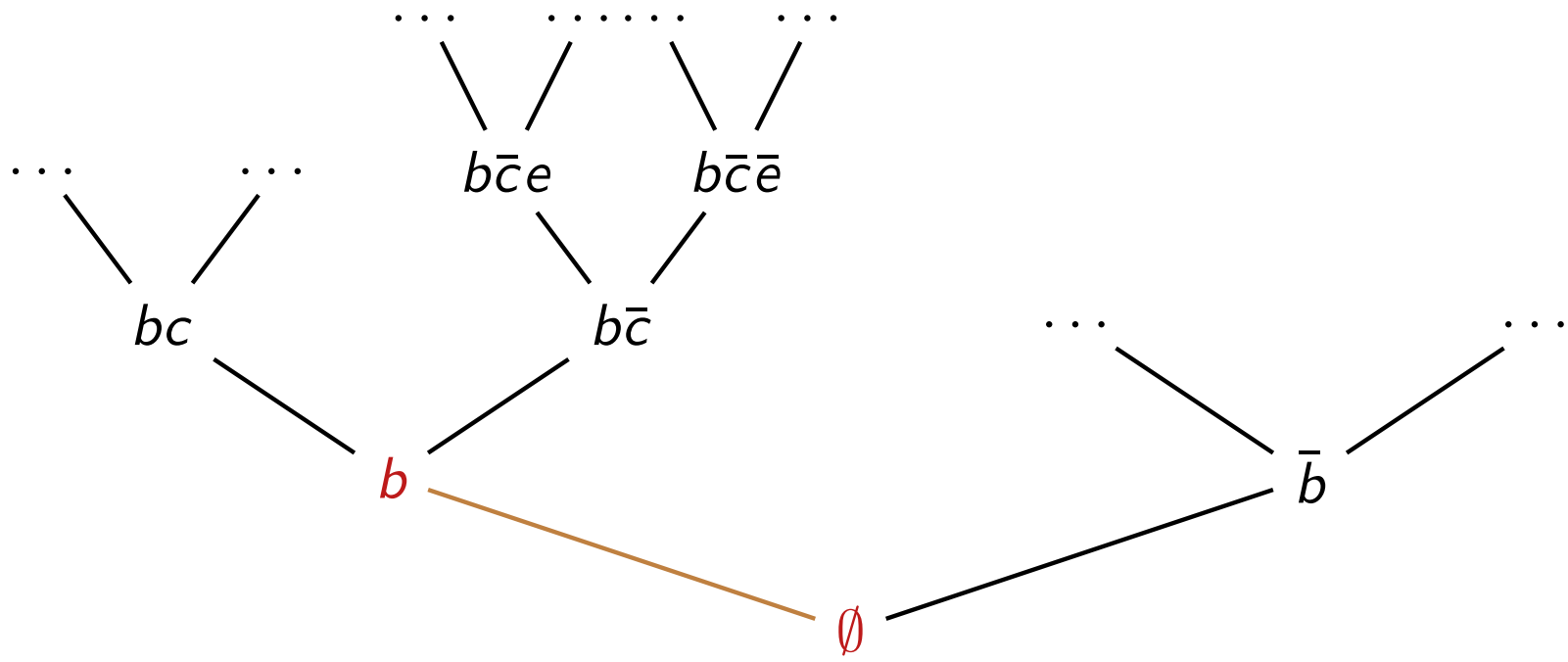
Resolution Refutation

Resolution Step: $\dfrac{A \vee x \quad B \vee \bar{x}}{A \vee B}$



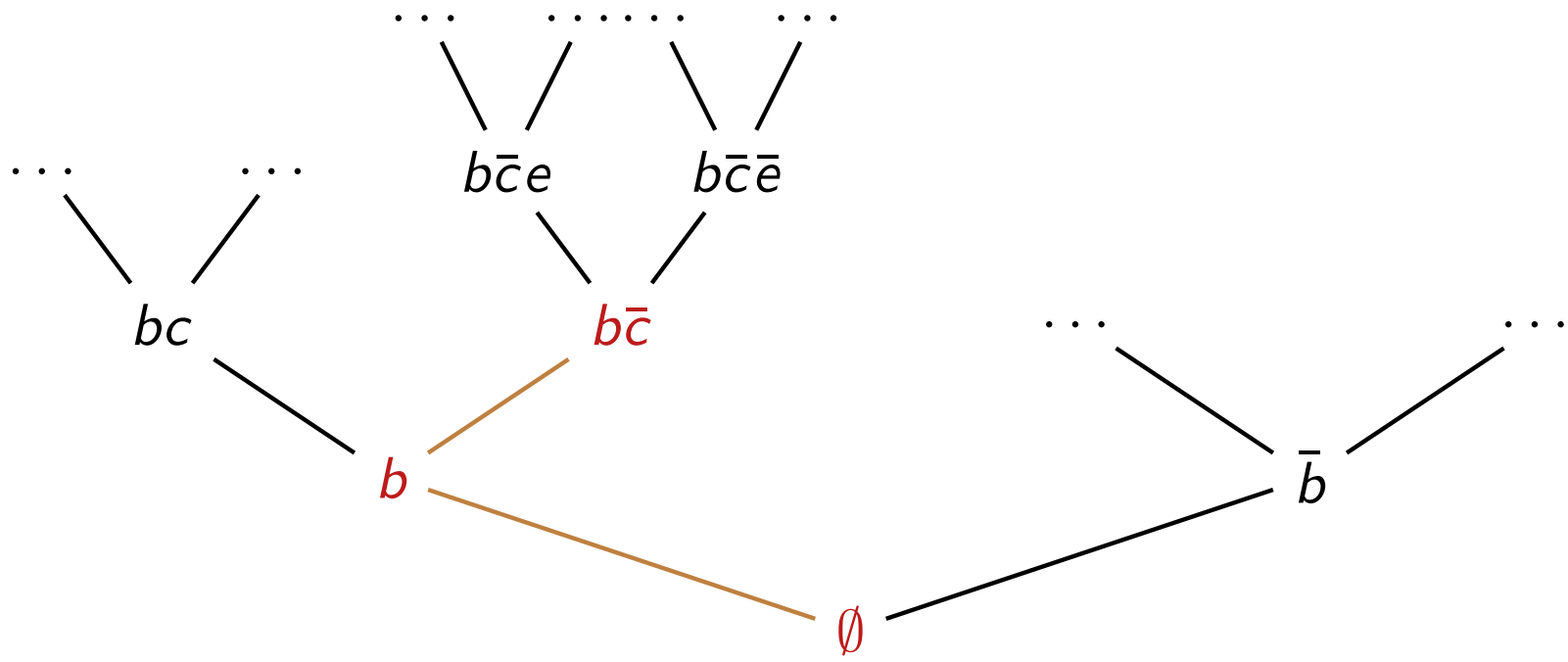
Resolution Refutation

Resolution Step: $\dfrac{A \vee x \quad B \vee \bar{x}}{A \vee B}$



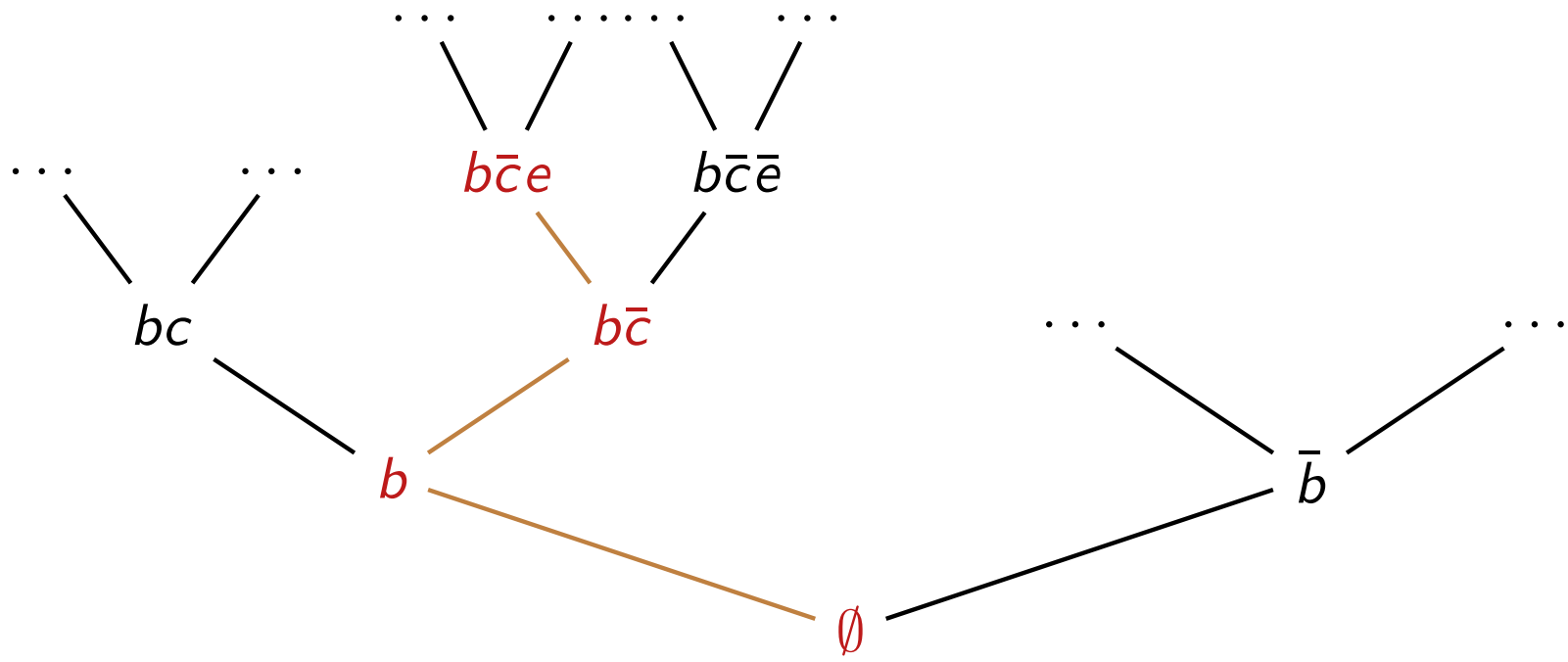
Resolution Refutation

Resolution Step: $\dfrac{A \vee x \quad B \vee \bar{x}}{A \vee B}$



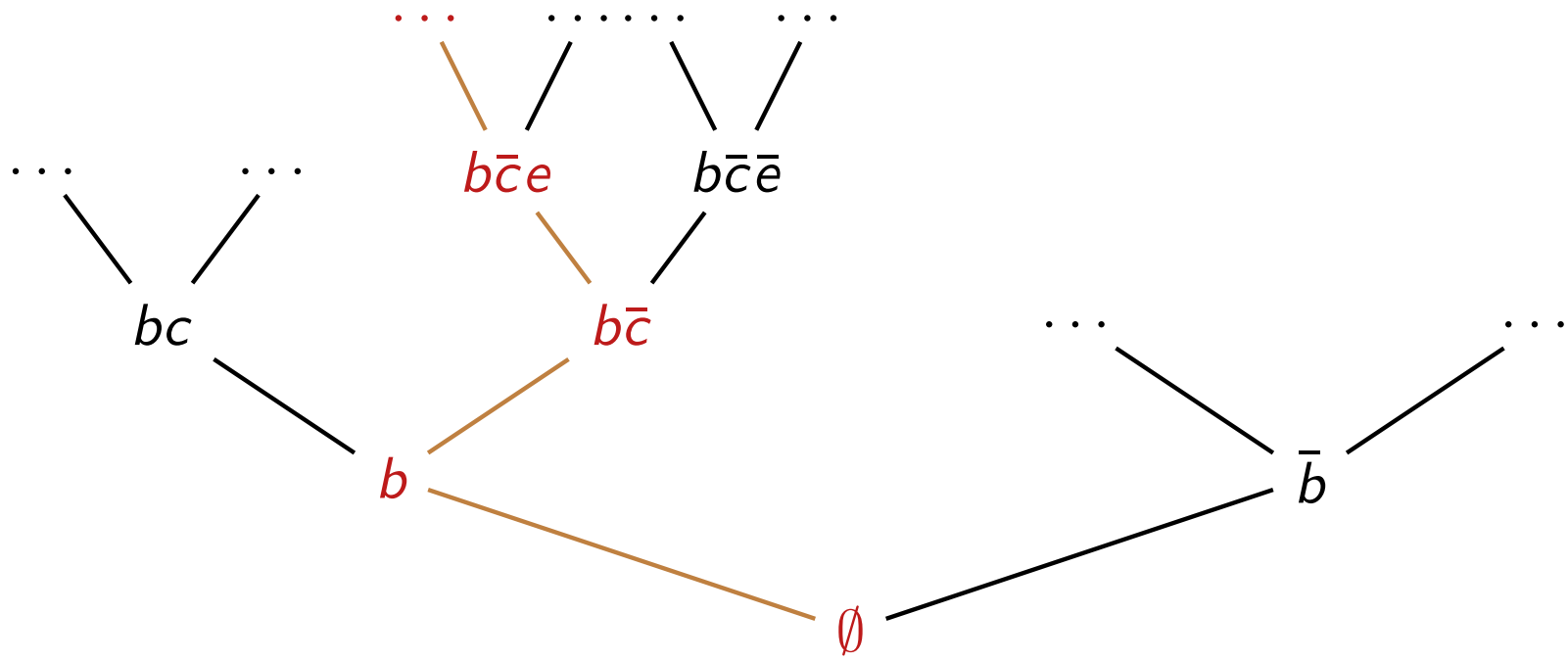
Resolution Refutation

Resolution Step: $\dfrac{A \vee x \quad B \vee \bar{x}}{A \vee B}$



Resolution Refutation

Resolution Step: $\dfrac{A \vee x \quad B \vee \bar{x}}{A \vee B}$



Partial Assignment on Resolution Refutation

- When a branch grows to a clause of Σ_G, this partial assignment falsifies the clause

Partial Assignment on Resolution Refutation

- When a branch grows to a clause of Σ_G, this partial assignment falsifies the clause
- If this partial assignment does not falsify any clause of Σ_G, then the branch must grow deeper!

Partial Assignment on Resolution Refutation

- When a branch grows to a clause of Σ_G, this partial assignment falsifies the clause
- If this partial assignment does not falsify any clause of Σ_G, then the branch must grow deeper!

- To falsify a clause from Σ_G:

Source	d	set d to $\textsc{False}$
Implication	$\bar{b} \vee \bar{c} \vee a$	set b, c to $\textsc{True}$, a to $\textsc{False}$
Sink	$\bar{a}$	set a to $\textsc{True}$

Partial Assignment on Resolution Refutation

- When a branch grows to a clause of Σ_G, this partial assignment falsifies the clause
- If this partial assignment does not falsify any clause of Σ_G, then the branch must grow deeper!

- To falsify a clause from Σ_G:

Source	d	set d to $\textsc{False}$
Implication	$\bar{b} \vee \bar{c} \vee a$	set b, c to $\textsc{True}$, a to $\textsc{False}$
Sink	$\bar{a}$	set a to $\textsc{True}$

- Adversary Argument:
  - When a variable is queried, answer $\textsc{True}$ or $\textsc{False}$

Partial Assignment on Resolution Refutation

- When a branch grows to a clause of Σ_G, this partial assignment falsifies the clause
- If this partial assignment does not falsify any clause of Σ_G, then the branch must grow deeper!

- To falsify a clause from Σ_G:

Source	d	set d to $\textsc{False}$
Implication	$\bar{b} \vee \bar{c} \vee a$	set b, c to $\textsc{True}$, a to $\textsc{False}$
Sink	$\bar{a}$	set a to $\textsc{True}$

- Adversary Argument:
  - When a variable is queried, answer $\textsc{True}$ or $\textsc{False}$
  - Try to avoid falsifying a clause from Σ_G (as above)

Partial Assignment on Resolution Refutation

- When a branch grows to a clause of Σ_G, this partial assignment falsifies the clause
- If this partial assignment does not falsify any clause of Σ_G, then the branch must grow deeper!

- To falsify a clause from Σ_G:

Source	d	set d to FALSE
Implication	$\bar{b} \vee \bar{c} \vee a$	set b, c to TRUE, a to FALSE
Sink	$\bar{a}$	set a to TRUE

- Adversary Argument:
  - When a variable is queried, answer TRUE or FALSE
  - Try to avoid falsifying a clause from Σ_G (as above)
  - Number of answers before falsifying $\leq$ depth of resolution refutation

Raz–McKenzie Pebble Game

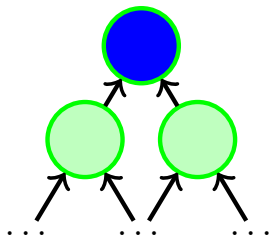
- Two players: **Pebbler** and **Colourer**, competitive

Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first

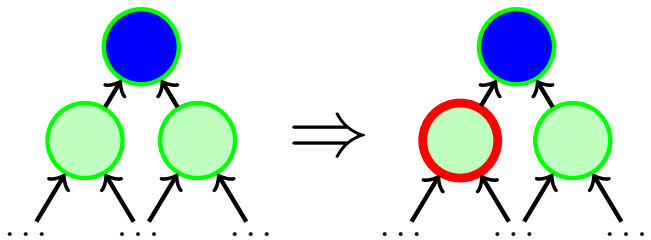
Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:



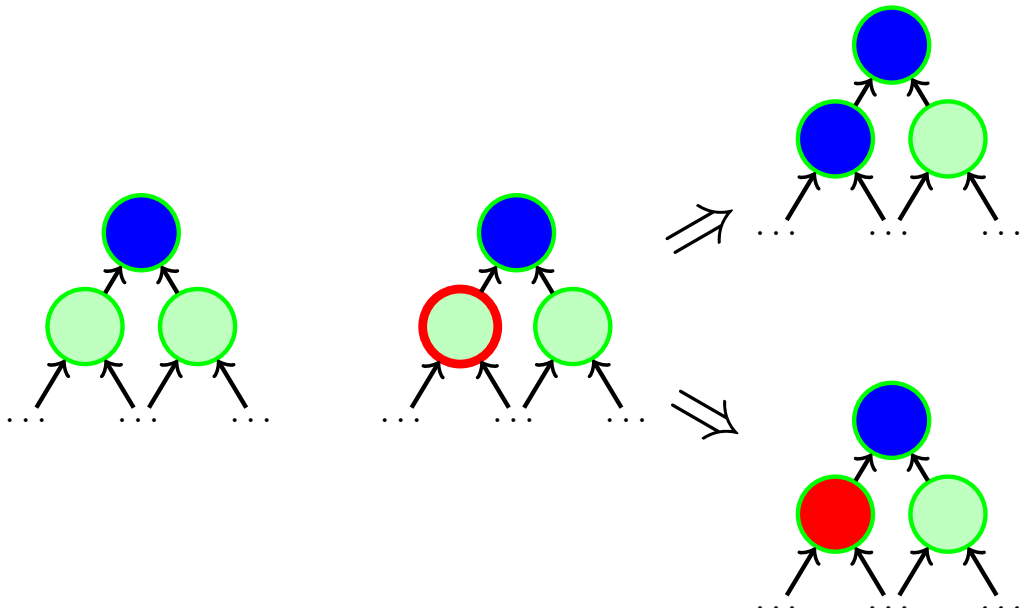
Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble



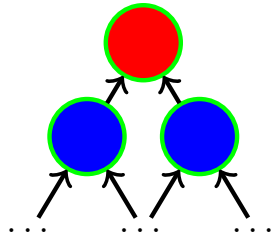
Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE



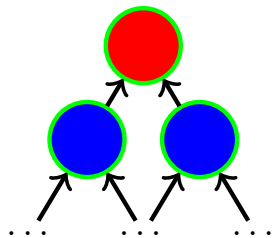
Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE (source and sink are treated analogously)



Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE (source and sink are treated analogously)
- **Colourer** aims to delay the inevitable



Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE (source and sink are treated analogously)
- **Colourer** aims to delay the inevitable

- Is exactly the depth of resolution refutation for Σ_G [C. '13]

Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE (source and sink are treated analogously)
- **Colourer** aims to delay the inevitable

- Is exactly the depth of resolution refutation for Σ_G [C. '13]
- **Colourer** strategy gives lower bound

Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE (source and sink are treated analogously)
- **Colourer** aims to delay the inevitable

- Is exactly the depth of resolution refutation for Σ_G [C. '13]
- **Colourer** strategy gives lower bound
- **Pebbler** strategy gives upper bound

Raz–McKenzie Pebble Game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE (source and sink are treated analogously)
- **Colourer** aims to delay the inevitable

- Add an initial set-up to make it more like Dymond–Tompa game.

Raz–McKenzie pebble game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Colourer** colours sink FALSE
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE
- **Colourer** aims to delay the inevitable

Raz–McKenzie pebble game

- Two players: **Pebbler** and **Colourer**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Colourer** colours sink FALSE
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Colourer** chooses to colour it TRUE or FALSE
- **Pebbler** wins if, before she moves, some FALSE node has all immediate predecessors TRUE
- **Colourer** aims to delay the inevitable

Dymond–Tompa pebble game

- Two players: **Pebbler** and **Challenger**, competitive
- Alternate to move, **Pebbler** moves first
- Initial Set-up:
  - **Pebbler** pebbles sink
  - **Challenger** challenges sink
- Each round:
  - **Pebbler** chooses a node to pebble
  - **Challenger** chooses to stay or jump
- **Pebbler** wins if, before she moves, the challenged node has all immediate predecessors pebbled
- **Challenger** aims to delay the inevitable

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):
  1. Turn a **Colourer** strategy (Raz–McKenzie game) into a **Challenger** strategy (Dymond–Tompa game).

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):
  1. Turn a **Colourer** strategy (Raz–McKenzie game) into a **Challenger** strategy (Dymond–Tompa game).
  2. If the Dymond–Tompa game is over, so is the Raz–McKenzie game.

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):
  1. Turn a **Colourer** strategy (Raz–McKenzie game) into a **Challenger** strategy (Dymond–Tompa game).
  2. If the Dymond–Tompa game is over, so is the Raz–McKenzie game.
  3. Implies Dymond–Tompa #Pebble $\geq$ Raz–McKenzie #Pebble.

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):
  1. Turn a **Colourer** strategy (Raz–McKenzie game) into a **Challenger** strategy (Dymond–Tompa game).
  2. If the Dymond–Tompa game is over, so is the Raz–McKenzie game.
  3. Implies Dymond–Tompa #Pebble $\geq$ Raz–McKenzie #Pebble.

Colourer strategy $\Rightarrow$ **Challenger** strategy:

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):
  1. Turn a **Colourer** strategy (Raz–McKenzie game) into a **Challenger** strategy (Dymond–Tompa game).
  2. If the Dymond–Tompa game is over, so is the Raz–McKenzie game.
  3. Implies Dymond–Tompa #Pebble $\geq$ Raz–McKenzie #Pebble.

Colourer strategy $\Rightarrow$ **Challenger** strategy:

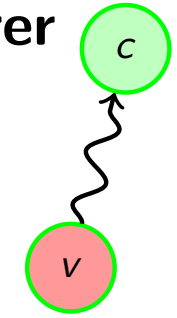
- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:

Dymond–Tompa Game = Raz–McKenzie Game

- Simulation argument (reduction in combinatorial game):
  1. Turn a **Colourer** strategy (Raz–McKenzie game) into a **Challenger** strategy (Dymond–Tompa game).
  2. If the Dymond–Tompa game is over, so is the Raz–McKenzie game.
  3. Implies Dymond–Tompa #Pebble $\geq$ Raz–McKenzie #Pebble.

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured FALSE
  - **stay** otherwise



Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured FALSE
  - **stay** otherwise

Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured $\textsc{False}$
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured $\textsc{False}$
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

- Invariant: challenged node c is the 'earliest' $\textsc{False}$ node.

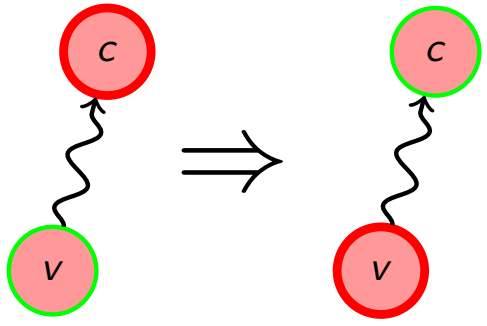
Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured $\textsc{False}$
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

- Invariant: challenged node c is the 'earliest' $\textsc{False}$ node.
  - Proof: by induction. $\square$



Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured $\textsc{False}$
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

- Invariant: challenged node c is the 'earliest' $\textsc{False}$ node.
  - Proof: by induction. $\square$
- When Dymond–Tompa game is over:

Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured $\textsc{False}$
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

- Invariant: challenged node c is the 'earliest' $\textsc{False}$ node.
  - Proof: by induction. $\square$
- When Dymond–Tompa game is over:
  - c is pebbled,
  - all immediate predecessors of c are pebbled,

Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured FALSE
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

- Invariant: challenged node c is the 'earliest' FALSE node.
  - Proof: by induction. $\square$
- When Dymond–Tompa game is over:
  - c is pebbled (FALSE),
  - all immediate predecessors of c are pebbled (TRUE),

Simulation Argument

Colourer strategy $\Rightarrow$ **Challenger** strategy:

- Assume c is challenged. If v is pebbled, see what a **Colourer** would do, and **Challenger**:
  - **jump** if v is a predecessor of c, and v is coloured $\textsc{False}$
  - **stay** otherwise

If Dymond–Tompa game is over, so is Raz–McKenzie game.

- Invariant: challenged node c is the 'earliest' $\textsc{False}$ node.
  - Proof: by induction. $\square$
- When Dymond–Tompa game is over:
  - c is pebbled ($\textsc{False}$),
  - all immediate predecessors of c are pebbled ($\textsc{True}$),
  - Raz–McKenzie game is over. $\square$

Summary of Results

- Equivalence of Pebble Games
  - Reversible Pebble Game
  - Dymond–Tompa Pebble Game
  - Raz–McKenzie Pebble Game

Summary of Results

- Equivalence of Pebble Games
  - Reversible Pebble Game
  - Dymond–Tompa Pebble Game
  - Raz–McKenzie Pebble Game
- Relations to Computational Complexity
  - Restricted lower bounds
  - Depth complexity of circuits

Summary of Results

- Equivalence of Pebble Games
  - Reversible Pebble Game
  - Dymond–Tompa Pebble Game
  - Raz–McKenzie Pebble Game
- Relations to Computational Complexity
  - Restricted lower bounds
  - Depth complexity of circuits
- Applications to Proof Complexity
  - Depth of resolution refutations
  - Size of Tree-Like resolution refutations

Summary of Results

- Equivalence of Pebble Games
  - Reversible Pebble Game
  - Dymond–Tompa Pebble Game
  - Raz–McKenzie Pebble Game
- Relations to Computational Complexity
  - Restricted lower bounds
  - Depth complexity of circuits
- Applications to Proof Complexity
  - Depth of resolution refutations
  - Size of Tree-Like resolution refutations
- Complexity of Pebble Games
  - PSPACE-complete

Summary of Results

- Equivalence of Pebble Games
  - Reversible Pebble Game
  - Dymond–Tompa Pebble Game
  - Raz–McKenzie Pebble Game
- Relations to Computational Complexity
  - Restricted lower bounds
  - Depth complexity of circuits
- Applications to Proof Complexity
  - Depth of resolution refutations
  - Size of Tree-Like resolution refutations
- Complexity of Pebble Games
  - PSPACE-complete (bounded fan-in)

Other Approaches

Other Approaches

Lower Bounds by Communication Complexity

Multi-party pointer jumping

[Chakrabarti '07] [Brody–Chakrabarti '08] [Viola–Wigderson '09]

$$\mathrm{ACC}^0 \stackrel{?}{=} \mathrm{P}$$

Other Approaches

Lower Bounds by Communication Complexity

Multi-party pointer jumping

[Chakrabarti '07] [Brody–Chakrabarti '08] [Viola–Wigderson '09] $\mathrm{ACC}^0 \stackrel{?}{=} \mathrm{P}$

Extensions of Karchmer–Wigderson framework

[Aaronson–Wigderson '09] $\mathrm{NL} \stackrel{?}{=} \mathrm{NP}$

[Kol–Raz '13] $\mathrm{NC} \stackrel{?}{=} \mathrm{P}$

Other Approaches

Lower Bounds by Communication Complexity

Multi-party pointer jumping

[Chakrabarti '07] [Brody–Chakrabarti '08] [Viola–Wigderson '09] $\mathsf{ACC}^0 \stackrel{?}{=} \mathsf{P}$

Extensions of Karchmer–Wigderson framework

[Aaronson–Wigderson '09] $\mathsf{NL} \stackrel{?}{=} \mathsf{NP}$

[Kol–Raz '13] $\mathsf{NC} \stackrel{?}{=} \mathsf{P}$

Size and Depth of Circuits

[Allender–Koucký '10] $\mathsf{TC}^0 \stackrel{?}{=} \mathsf{NC}^1$

[Lipton–Williams '12] $\mathsf{NC} \stackrel{?}{=} \mathsf{P}$

Other Approaches

Lower Bounds by Communication Complexity

Multi-party pointer jumping

[Chakrabarti '07] [Brody–Chakrabarti '08] [Viola–Wigderson '09]

$$\mathrm{ACC}^0 \stackrel{?}{=} \mathrm{P}$$

Extensions of Karchmer–Wigderson framework

[Aaronson–Wigderson '09]

$$\mathrm{NL} \stackrel{?}{=} \mathrm{NP}$$

[Kol–Raz '13]

$$\mathrm{NC} \stackrel{?}{=} \mathrm{P}$$

Size and Depth of Circuits

[Allender–Koucký '10]

$$\mathrm{TC}^0 \stackrel{?}{=} \mathrm{NC}^1$$

[Lipton–Williams '12]

$$\mathrm{NC} \stackrel{?}{=} \mathrm{P}$$

Geometric Complexity Theory

[Mulmuley–Sohoni '01 '08]

$$\mathrm{VP} \stackrel{?}{=} \mathrm{VNP}$$

Questions